RESPONSE TIME ANALYSIS OF NETWORK PERFORMANCE

INVENTORS

Steven J. Schaffer, Lone Tree, CO, USA

Jacob Weil, Palo Alto, CA, USA

COPYRIGHT NOTICE

FIELD OF THE INVENTION

[0002] The present invention relates to the field of network computing, and more particularly, to a method and system for monitoring network, client, and server performance.

BACKGROUND OF THE INVENTION

[0003] One method of monitoring network performance is by measuring the processing time on a first node, such as a client, and the processing time on a second node, such as a server. In conventional approaches, this approach was applied where the client and server were on the same local area network (LAN), so that factors such as network delay external to the LAN did not need to be considered.

[0004] Realistic implementations involve networks that generally include multiple LANS and interconnecting equipment and/or communications links. Thus, there is a need for an improved system and method of monitoring network performance whereby network delay is considered when monitoring network performance.

SUMMARY OF THE INVENTION

In one aspect of the invention, a method for monitoring network performance is disclosed. The method monitors a flow having one or more frames within a thread by calculating a node's active time. This includes the amount of time each frame is processed on a sending node in a network, the amount of time each frame is processed on a receiving node in the network, and the amount of time each frame is in transit across the network.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0006] FIG. 1 illustrates an Application Performance Report in a Chart format.

[0007] FIG. 2 illustrates an Application Performance Report in a Details format.

[0008] FIG. 3 illustrates an example of Request Preparation and Reply Preparation processing types.

[0009] FIG. 4 illustrates one exemplary embodiment.

[0010] FIG. 5 illustrates an example of a flow.

[0011] FIG. 6 illustrates an example of a multi-tier algorithm.

DETAILED DESCRIPTION OF THE INVENTION

[0012] The present invention includes various operations, which will be described below. These operations may be performed by hardware components or may be embodied in machine-executable instructions, which in turn may be used to cause a general-purpose or special-purpose processor or logic circuits programmed with the instructions to perform the operations. Alternatively, the operations may be performed by a combination of hardware and software.

[0013] The present invention may be provided as a computer program product that may include a machine-readable medium having stored thereon instructions which may be used to program a computer (or other electronic devices) to perform a process according to the present invention. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs (Compact Disc-Read Only Memories), and magneto-optical disks, ROMs (Read Only Memories), RAMs (Random Access Memories), EPROMs (Erasable Programmable Read Only Memories), EEPROMs (Electromagnetic Erasable Programmable Read Only Memories), magnetic or optical cards, flash memory, or other type of media / machine-readable medium suitable for storing electronic instructions. Moreover, the present invention may also be downloaded as a computer program product, wherein the program may be transferred from a remote computer (e.g., a server) to a requesting computer (e.g., a client) by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection). Accordingly, herein, a carrier wave shall be regarded as comprising a machine-readable medium.

Introduction

[0014] Response Time Analysis (hereinafter RTA) first produces underlying data. This data is then formatted and presented to a user in several reports, including the Application Performance Report (Chart and Details), Node Processing Time Report, and Flows Report. The reports are presented in the form of a Graphical User Interface (GUI).

[0015] The user's first exposure to the response time analysis is in the high-level summaries presented in the health report. The summaries enable the user to quickly obtain critical information without needing to process details. Details are made available in the Node Processing Time report and Flows Report.

[0016] The following sections give a thorough description of the algorithms and techniques used in the underlying RTA, as well as examples of the GUI.

Application Performance Report

Chart

[0017] FIG. 1 illustrates an Application Performance Report. The four summary results shown in the first panel of the application performance report are given in the following paragraphs.

Network Busy Time

[0018] The Network Busy Time 100 is the total time that one or more meaningful frames are in transit across the network. Network Busy Time is computed and reported for both network directions (from primary to secondary and vice-versa).

After the Flow concept has been introduced the Network Busy Time will be revisited to describe which frames are meaningful. Because only a subset of all frames generated traverse the network, only frames that are exchanged between the two capture points are included in the Network Busy Time. Consequently, Network Busy Time is applicable only with a multi-segment merged or adjusted trace. A trace is a sequence of frames that have flowed between two or more nodes over the capture period (to be discussed below). Traces can be merged and adjusted. The Network Busy Time can be broken down into insertion time and queuing propagation and processing time.

[0019] Insertion Time (sec): This is the cumulative time for the frames to be inserted into the network. In a merged or adjusted trace, the insertion time for each frame is computed as AdjustedBytes * 8 / Bandwidth. The network bandwidth utilization is computed for each direction of the network. Only frames that are known to have traversed the network are included. The term AdjustedBytes is used in the above expression to indicate the bytes that would have traversed the WAN link. The capture environment allows the user to specify whether the frame size should be adjusted to compensate for the different WAN headers. If the user chooses to do so, then AdjustedBytes = Bytes(as captured) – DLCHeader(as captured) + DLCHeader (specified in capture environment).

[0020] Network Queuing, Propagation & Processing (QPP) Time (sec) This is the time that frames were present in the network due to queuing (in routers), processing and propagation. This represents the portion of the total transit time that is not counted as insertion time. Throughout the description, this term is referred to as QPP time.

Node Active Time

[0021] Referring again to Fig. 1, in the Node Active Time area 102, there is one bar for each node that shows the active (processing or sending) duration. Each bar is broken down into the following two components.

[0022] **Node Processing Time** For each node, the overall task processing time is shown. A task is a user-invoked operation that creates network traffic during execution and causes a screen update or other acknowledgement on the user's machine. Once invoked, a task executes to completion without further user intervention.

[0023] For single-thread applications, this is simply the sum of the individual node processing times (described below). If an application can have multiple server requests outstanding (such as the typical Web browser), then the node is considered processing if it is handling one or more requests. The Node Processing Time for any node cannot exceed the task duration, whereas the sum of the Processing Times for all nodes can exceed the task duration if there are parallel (overlapping) threads.

[0024] **Node Sending Time** For each node, the overall node sending time represents the period during which the node is sending data but not otherwise processing. If a node is in the process of sending a set of frames, then the node is considered to be in the sending state, but only if it is not processing another request at the same time. Sending time is important because it could indicate that the node is processing in order to prepare the remaining frames, or otherwise not processing. The most likely other factor is that the network is heavily utilized and the node cannot send

all data in one transaction. Other potential scenarios include insufficient TCP window size, the normal slow-start nature of TCP, inability of the receiving node to remove the data from the TCP buffer quickly enough, or an inefficient TCP implementation at either the sending or receiving node.

[0025] Referring again to FIG. 1, the following detailed reports are available by double-clicking certain areas as discussed in the following paragraphs.

[0026] On a Node Processing Time portion of a node bar: Brings up the Node Processing Detail report filtered / highlighted on the specified node.

[0027] On a Node Sending Time portion of a node bar: Brings up the Node Sending Detail report filtered / highlighted on messages sent by the specified node.

[0028] On either network bar 100: Brings up the Network Utilization and Latency graph.

Details

[0029] An example of an Application Performance Report — Details is shown in FIG. 2. The major sections of the detailed report are described in the following sections.

Overall Summary

[0030] The overall summary 200 provides information on the task duration, any errors that were detected, the capture environment, and a summary of the conversations, threads and turns. The information in the overall summary can alert the

user to errors, or to the fact that other than the desired information was captured. The values displayed in this section are as follows.

[0031] **Task Time** (sec) This is the duration of the task, and should be equivalent to the task "stopwatch time." If this is not the case, then portions of the time may be missing or may need to be deleted.

[0032] **Traffic Duration** (sec) This is the duration within which frames exist. The user can click on the label to open the bounce diagram, which will show all constituent frames and their durations.

[0033] **Errors** A click on the Errors label opens the Error Report, which is a graphical summary of the number and types of errors and warnings detected in the trace.

[0034] **Capture Environment** This legend indicates the meaning of the arrows shown in many other places within the report. A capture is a process whereby a traffic collector or agent collects network frames exchanged between nodes in a distributed application and stores the data for off-line analysis. The legend keys are taken from the Capture Environment as specified by the user. The primary capture location is indicated on the left and the secondary location is on the right.

[0035] **Conv** indicates Conversations, both for the task and the total, and those that occur between a node on the primary location and another node in the secondary location, as indicated by the "<-->" label. Hereafter, the phrase "conversations that traverse the network" means conversations for which one node is in the primary location and the other is in the secondary location.

[0036] **Threads** This shows both the total number of threads in the task as well as in the conversations that traverse the network between the primary and secondary capture points. A thread is a sequence of frames exchanged between two nodes that constitutes a single application or protocol action. For example, the retrieval of a graphic from a WWW (World Wide Web) server is a thread.

[0037] **Turns** This is the number of turns in the task, and in the conversations that traverse the network between the primary and secondary capture points. <--> Turns specifies the sum of the turns for threads corresponding to one node in the primary location and another other node in the secondary location.

[0038] **Bytes/Turn** This is the average number of bytes per turn, both as a total for the task and for the conversations that traverse the network.

[0039] **Frames/Turn** (not shown) This is the average number of frames in each turn, both as a total for the task and for the conversations that traverse the network.

Traffic

[0040] Traffic section 202 provides the user with a summary of several traffic measures, for the entire task (Total row), over the network (<--> row, i.e., in both directions between the primary and secondary capture points), and in each direction across the network. Columns in this section are discussed in the following paragraphs.

[0041] **Bytes** This is the sum of the bytes for all frames in each classification. For the network classifications, the byte counts for each frame will be

adjusted for the DLC header size if the user has chosen to do so in the capture environment.

[0042] For <-->, → and ← Bytes, the value is the number of bytes that crossed the point of contention in the network as specified for the task in the Capture Environment. If the user did not choose to adjust frames for DLC header in the capture environment, then Network Bytes = Bytes for each frame. If the user did so choose, then Network Bytes = Bytes – DLC Header(as captured) + DLCHeaderbytes (specified in the capture environment).

[0043] **% of <-->** Bytes This column shows the percentage of <--> bytes that traversed the network in each direction. The two values will add to 100% (subject to rounding).

[0044] **Frames** This is the total number of frames in the task (top row), and the number of frames that should have crossed the network, whether they were contained in both captures or not. If they were not, such will be reported in the two Frames Missing entries to the right of this section. A frame is a collection of bits comprising data and control information (overhead) that is transmitted between nodes in a network.

[0045] **Avg Frame** This is the average frame size, in bytes, i.e., Bytes / Frames.

[0046] **Captured Load** (kbps and %) This is the average rate, in kbps and as a percentage of the total bandwidth, of the frames that traversed the network in each direction. The adjusted bytes, as discussed above, are used.

[0047] **Frames Missing at Source** This is the number of additional frames from the sending (source) node that should have been in the trace. This can be caused by the capture beginning too late or ending too early, or by the inability of the capture device to capture all of the frames.

[0048] **Frames Missing at Destination** (not shown) This is the number of additional frames that should have been in the trace from the receiving (destination) node. Such frames could have been lost due to network congestion or failure of a network component, or for the reasons discussed in the preceding paragraph.

[0049] There can be other situations wherein not all of the frames that should have traversed the network were actually captured. For example, one of the captures may have started before or ended after the other and may contain frames that traversed the network. However, since those frames are not represented in the other capture, it is not known if they actually traversed the network. Such frames will be flagged Lost Frame or Dropped Frame, depending on whether they were captured only at the source or destination segment, respectively.

[0050] If it is assumed that missing frames really did traverse the network, then their bytes / frames / threads / conversations are included in the <--> metrics. There is no way to know whether a frame that is missing at the destination actually consumed bandwidth at the network contention point before being dropped. Thus, the worst case is assumed, i.e., that network bandwidth was consumed, and the missing frames are included in the <--> metrics.

Network Busy Time

[0051] The Network Busy Time section 204 helps the user determine how active the network was during the task, and how the busy time breaks down into insertion time and QPP time.

[0052] The Network Busy Time section 204 is presented for merged tasks and single-trace adjusted tasks. There are two rows for the network metrics, one for the primary to the secondary location (→), and the second from the secondary to the primary location (←). The columns in this section correspond exactly to the portions of the network bars 100 in the Application Performance Report chart of FIG. 1.

[0053] **Insertion Time** (sec and % of Task Time) This information represents the cumulative time that it took to insert the captured frames into the network at the point of lowest bandwidth specified by the user. Should the user decide to adjust for DLC headers of the network by the captured bytes, the insertion time is based on the network bytes of each frame. The bandwidth utilization in kbps is determined as (Bytes that traversed the network in the specified direction * 8) / (duration of the task in seconds * 1000). The bandwidth utilization in % is (the bandwidth utilization in kbps / capacity of the link in kbps), expressed as a percentage. The capacity of the link is specified by the user in the capture environment.

[0054] **QPP Time** (sec and % of Task Time) This is the Queuing, Processing and Propagation time.

[0055] **Total** (sec and % of Task Time): The total time that one or more meaningful frames was traversing the network. Meaning frames include all data frames

and TCP acknowledgements that are within the data portion of a message. Meaningful frames do not include TCP acknowledgements that are sent after the last data frame in a message is sent.

Network Frame Transit Statistics

[0056] The network frame transit statistics section 206 comprises 2 rows, one for each network direction as described above. This section includes only meaningful frames. As such, the statistics do not include TCP acknowledgements that are sent after the last data frame in a message is sent.

[0057] **Transit Time** (sec) This column will reflect the graphical depiction of the minimum, average and maximum frame transit time. The transit time of each frame is the difference between its Time Sent and Time Received.

[0058] **Transit Time Min** (sec) This is the transit time of the frame that has the lowest transit time.

[0059] **Transit Time Avg** (sec) This is the average (mean) transit time for the meaningful frames.

[0060] **Transit Time Max** (sec): This is the transit time of the frame that has the largest transit time.

[0061] **Transit Time Frame Count** This is the number of frames included in the transit time statistics, and is equal to the number of meaningful frames that were captured at both sides (or adjusted). Note that this number is equal to or less than the number of frames that traversed the network in the specified direction. It does not

include TCP acknowledgements after a message or frames that are missing at the source or destination.

[0062] **Latency Statistics** (min, avg, max, not shown) These are statistics on the latency of meaningful frames that traverse the network. As described above, meaningful frames are data frames and the TCP acknowledgements that occur during the data transfer portion of a flow.

[0063] **Overlap Avg** (Frame) This is the average number of frames that are in transit when there is at least one frame in transit. It is a measure of the application's ability to send more than one frame at a time, and the network conditions requiring the application to do so. In other words, it is the average number of frames sent by the application when the application has sent frames. Higher values mean the application is less susceptible to network latency and bandwidth.

[0064] **Overlap Max** (Frame) This is the largest number of frames that are in transit in the network at any given time.

Node Active Time

[0065] This is a tabulation 208 of the node bars 102 (Node Processing And Sending times) that were described earlier in the Application Performance Report Chart of FIG. 1.

Node Processing Statistics

[0066] Statistics 210 regard the node processing periods, as discussed in the following paragraphs.

[0067] **Processing Time** (sec) This reflects a graphical depiction of the minimum, average and maximum node processing time period.

[0068] **Processing Min** (sec) This is the shortest node processing period.

[0069] **Processing Avg** (sec) This is the average node processing period.

[0070] **Processing Max** (sec) This is the longest node processing period.

[0071] **Processing Periods** This is the number of processing periods, and is important as it tells the user if there is a large or small number of node processing periods.

[0072] **Overlap Avg** This is the average number of processing periods that occur simultaneously at the node during the times that the node is in at least one processing period. A value of 1.0 means that the node never processed more than one request at a time. This value cannot be smaller than 1.0.

[0073] **Overlap Max** This is the largest number of processing periods occurring at any given instant.

Node Processing Detail Report

[0074] In addition to the overall summary results presented in the Application Performance Report, the RTA also identifies and reports several sets of details. One of these is the node processing detail (not shown).

[0075] Each node processing time is one component in the Overall Node Processing Time. The GUI allows the operator to see the Individual Node Processing

Times for all nodes or for one node at a time. Note that since individual node processing times can overlap, the sum of the individual node processing times can be greater than the Overall Node Processing Time for a given node. The attributes of each node processing time, as given in the columns in the Node Processing Detail Report are listed or described in the following sections.

[0076] **Node Name**

[0077] **Node Address**

[0078] **Errors** This is the number of errors associated with either the start or the end frame. The user can click on the error report to see the individual errors.

[0079] **Duration** This is the time span of the processing time, in seconds.

[0080] **Start Time** This is the time at which the node began processing

[0081] **Start Frame** This is the number of the frame captured at the beginning of the processing time. The user can click on this parameter to view the bounce or packet trace as of the start frame.

[0082] **End Time** This is the time at which the node stopped processing.

[0083] **End Frame** This is the number of the frame captured at the end of the processing time. The user can click on this parameter to view the bounce or packet trace at the end of the frame.

[0084] **Start Frame Description** This is the description (decoded) of the start frame.

[0085] **End Frame Description** This is the description (decoded) of the end frame.

[0086] **Node Processing Type** This is one of the types specified in the sections below. For most node processing types, there are corresponding types for client and server. "Client" is assigned when the node is the client in a thread, and vice versa. Each node processing type has an internal code that does not appear in the GUI. The node processing types and their meanings are given in the following sections.

[0087] **Client Before Thread** This is shown in FIG. 3 at 300, and represents the time period prior to sending of the first data frame by the client. The period extends back to the previous data frame that was received by the node, or to the beginning of the task if there is no such frame.

[0088] **Client Processing** This the period within a thread from receipt of a data frame by the client node to the time that node sends a subsequent request. The data frame can be on the same thread or another thread.

[0089] **After last frame** 308 This is the time period from the last data frame to the end of the task, and is always assigned to the client node for that task. The client node can be reassigned in the conversation map.

[0090] **Server Before Thread** This is similar to Client Before Thread, but arises when the first data frame in the thread is sent by the server of the thread. Normally the server does not send the first data frame, since the client normally initiates activity by sending a request. However, if the capture starts in the middle of a thread, or if the client and server are assigned incorrectly then, then this processing type results.

[0091] Note that the Thread Analysis window comprises a command to swap the client and server of a thread if they are identified incorrectly.

[0092] **Server Processing** 304 This is the period within a thread from receipt of the last frame in a request by the server to the time that the first response frame is returned by the server. During this time, the server is assumed to be processing the request. Note that with multi-tier applications (discussed below) there are cases where an upper-tier request may interrupt a server processing time.

[0093] **Request Preparation** 302 This processing type arises in multi-tier applications. It is the period from receipt of a request by a mid-level server (from a lower tier) until the mid-level server begins sending a subsequent request to another server.

[0094] **Reply Preparation** 306 This processing type also arises in multi-tier applications. It is the period from receipt of a reply by a mid-level server (from another server) to initiation of a reply to the requesting node.

Flows Report

[0095] A flow is a set of data frames that is sent from one node to another, comprises only frames in a single thread, and spans a time period during which no data frames travel in the opposite direction. A flow includes the TCP acknowledgements that are sent in the opposite direction before the transmission direction reverses.

[0096] Meaningful frames, discussed above, comprise all data frames and TCP acknowledgements within a data flow. TCP acknowledgements that occur after the last data frame in a flow are not meaningful frames for the purpose of network busy time computation.

[0097] In the flows report, each flow includes a number of attributes, arranged in columns, as discussed in the following sections.

[0098] **Errors** The flows report provides a graphical depiction of relevant errors and warnings, as with other reports. A link to the error report is provided. Since a flow comprises one or more frames, the errors associated with any frame in the flow should be included in this summary.

[0099] **Sending Node** This is the name and address of the node that sent the data frames in the flow. This node will receive TCP acknowledgements from the corresponding receiving node.

[0100] **Receiving Node** This is the name and address of the node that received the data frames in the flow. This node will send TCP acknowledgements to the corresponding sending node.

[0101] **Data Duration** This is the period in seconds from the time the sending node sent the first frame in the flow to the time that the receiving node received the last data frame in the flow. This is related to other fields by the relationship Data Duration = (End Data Time - Start Time)

[0102] **Avg Data Rate** This is the average data rate in bits/sec during the flow. This information may be important to the user, because flows with low data rate may be demand investigation. For example, the user may need to determine why the data is not being transferred more quickly, particularly if the flow is also longer than most other flows. This is computed as (Data Payload Bytes * 8) / (End Data Time — Start Time).

[0103] **Bytes** This is the total number of bytes in all frames in the flow.

[0104] **Data Payload Bytes** This is the sum of the payload bytes in all frames in the flow.

[0105] **Frames** This is the number of frames in the flow.

[0106] **Data Frames** This is the number of data frames in the flow.

[0107] **First Frame** This is the sequence number of the first frame in the flow.

[0108] **Last Data Frame** This is the sequence number of the last data frame in the flow.

[0109] **Last Frame** This is the sequence number of the last frame, either data or acknowledgement. If there are TCP acknowledgement frames after the last data frame, then this will differ from the Last Data Frame.

[0110] **Start Time** This is the time that the first data frame was sent.

[0111] **End Data Time** This is the time that the last data frame was received.

[0112] **End Time** This is the time that the last frame (data or acknowledgement) was received. Note that this is normally not important because trailing TCP acknowledgements do not have an impact on the response time.

[0113] **Data Direction** This reflects primary-to-secondary direction or vice versa, as indicated with (→ and ←) arrows. For flows that are within a capture location,

the caption "within <capture point>" appears, where <capture point> indicates the location of interest.

[0114] **Network Busy Time** This is the total time in seconds that one or more frames was in transit during the flow.

RTA Algorithm Details

Overall Approach

[0115] The RTA functions first at the thread level. Each thread is assumed to be a single-threaded sequence of request / response exchanges between the client and server. It is the responsibility of the protocol decoder to ensure this requirement. A thread is broken down into 5 time periods, described in the following sections.

[0116] Client Processing

[0117] **Client Sending** Flow is being sent from client to server

[0118] Server Processing

[0119] **Server Sending** Flow is being sent from server to client

[0120] **Processing interrupted by another thread** This period only occurs in the case of multi-tier or overlapping requests at the client

Exemplary Embodiment - Single Thread

[0121] RTA concepts are illustrated according to one embodiment as shown in FIG. 4. FIG. 4 is a bounce diagram for a typical client / server or Web

application. The application could be, e.g., a 2-tier SQL application, a web browser / web server, or an application based on ad hoc protocols.

[0122] Assume that all frames shown exist within a single thread. In this example, the client sends a 2-frame request to the server, the server processes over a period, and then the server sends a 3-frame reply to the client. The diagram shows the data frames that would be exchanged, as well as exemplary TCP acknowledgements if the application uses TCP/IP. Note that TCP is a very dynamic protocol, and therefore may not send a TCP acknowledgement for every frame. Accordingly, the diagram illustrates only one of many possible variations.

Node Processing and Sending

[0123] The Application Performance Report Chart of FIG. 1 would show the processing time for the client as the sum of the two processing times identified in FIG. 4, i.e., one at the beginning and one at the end. The processing time for the server is just the single processing time 304, and the sending time for both nodes is as indicated in the FIG. 4.

Flows

[0124] In this example there are two flows. The first flow is from the client to the server and comprises frames 1 through 4. The last data frame in this flow is frame 3. The overall flow duration and flow data duration are annotated in FIG. 5. Because Frame 4 is a TCP acknowledgement that is sent after the last data frame in the flow (Frame 3) is sent, it is not considered a meaningful frame. Thus, the network is not considered busy for the time that frame 4 is in transit.

[0125] Similar analysis applies to the second flow in this example. The second flow is sent from the server to the client, and comprises frames 5 through 10. The data duration for this flow spans the time frame 5 is sent to the time frame 8 is received.

Network Frame in Transit and Latency Statistics

[0126] Again referring to FIG. 4, the times when a frame is in transit can be seen. In general, the TCP acknowledgements that are returned after data is completely received have no impact on the overall response time; therefore, they are omitted from the network Frame in Transit measure and Latency statistics. Accordingly, in this example frames 4 and 10 do not impact the user-perceived application response time, and thus they are not included in the network latency measures.

[0127] A high network-frame-in-transit time can be an indication that the combination of network latency and the application's sequential request / response behavior is affecting the response time. A high network busy time may be caused by insufficient network bandwidth. This may be investigated by consulting the network utilization information in the Performance Report Chart. This chart breaks down a network frame in transit into components caused by bandwidth and latency. If the bandwidth component dominates, then lack of bandwidth is causing the network to be busy.

Exemplary Embodiment - Multi-Thread

Node Processing and Sending Time

[0128] Node processing time analysis becomes more complicated when

the application is multi-threaded. In the example above, the sum of the node processing times equals the node active times. When node processing times overlap at a node, only one such time is counted; consequently, the sum of the individual node processing times can be greater than the calculated overall node processing time.

[0129] The user is shown results for both node processing and sending times. Node processing time unambiguously reflects periods during which the node is processing requests (as a server) or processing a reply prior to sending the next request (as a client). Conversely, node sending time reflects not only node processing, but potentially other activity as well. It is difficult to determine what contributed to the node sending time without examining the individual flows sent. For example, node sending times could result from factors such as the receiving node's inability to process incoming data quickly enough, insufficient bandwidth in the network, the sending node's inability to make all of the data available quickly enough, or a TCP window size that is too small. Consequently, node sending time might contribute to the total response time. For example, in cases where the network is heavily utilized or has high latency, a high node sending time can be caused by limitations of the network. To explore this, a user can link from the node-sending-time bar to view the flows sent by that node. The user could thereby attempt to identify whether the lapse between frames sent by the node represented actual processing time. Often this determination will require some knowledge of the application.

Node Processing Times

[0130] The Node Processing Detail report (not shown) shows the processing times that were detected for each node. The report is arranged in order of

descending duration, with the largest processing times at the top. Processing time at a client node begins just prior to sending the first request in a thread, and ends within a thread prior to each succeeding request that the node sends. Processing time at a server node begins when the server receives a request and ends when the server begins sending a reply.

[0131] For further understanding of node processing times, a Node Processing Detail report can be opened on a trace. The window can then be split, and the packet trace placed at the bottom. For each node processing time selected, there will be two frames surrounding the processing time. For client processing times, there will be a prior reply (or the beginning of the trace) and the request that the client sends at the end of its processing time. For server processing times, there will be the request (actually the last frame in the request if it is a multi-frame request) and the response (actually the first frame in the response if it is a multi-frame response).

Flows

[0132] As discussed previously, there can be many causes for a high node sending time, and the cause of this can be difficult to determine.

Overlapping Threads

[0133] Any time two or more threads overlap, more than one node can be processing or sending at a time. Alternately, a single node could be processing or sending on more than one thread at the same time. These processing and sending times are aggregated by concluding that a node is processing if it is processing on one or more threads, or that a node is sending if it is sending on one or more threads and is

not processing.

Exemplary Embodiment - Multi-Tier

[0134] FIG. 6 describes the handling of multi-tier applications, as follows. At each time that a data frame enters a node, the time, frame seq# and its thread is recorded in member variables of the CNode class. When a client of a node sends out the first frame in a request, there will be a processing time. To determine its type, it is determined whether the most recent data frame that arrived at the node was a request frame from another client. If it was, then the type is 'Request Preparation,' otherwise, the type is 'Client Before Thread.'

Conclusion

[0135] A method and system for performing response time analysis of network performance have been described. Although the present invention has been described with reference to specific exemplary embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the spirit and scope of the invention. Accordingly, the above description and drawings are to be regarded in an illustrative rather than a restrictive sense.